RVA-clustering: An Approximation-based Indexing Approach for Multi-dimensional Objects

Cristian-Augustin Saita, François Llirbat

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

RVA-clustering: An Approximation-based Indexing Approach for Multi-dimensional Objects

Cristian-Augustin Saita — François Llirbat

N° 4670

Decembre 2002

THÈME 3

ISSN 0249-6399 ISRN INRIA/RR--4670--FR+ENG

RVA-clustering: An Approximation-based Indexing Approach for Multi-dimensional Objects

Cristian-Augustin Saita , François Llirbat

Thème 3 — Interaction homme-machine,
images, données, connaissances
Projet Caravel

Rapport de recherche n° 4670 — Decembre 2002 — 29 pages

Abstract: In this paper we propose a new approach for efficiently answering spatial queries (intersections, containments, enclosures) over large databases of multi-dimensional objects (hypercubes). A wide range of applications could benefit of our technique: image retrieval, document indexing, time series, notification systems, and other applications involving multi-dimensional spatial data. Our contribution consists in the definition of an approximation model for multi-dimensional objects and spatial operations, which accelerates the object verification and enables a database organization in clusters, to avoid the exhaustive database scan. The grouping strategy based on access probabilities allows the clustering to behave efficiently against skewed data and/or skewed queries. Performance analysis shows that our approach efficiently copes with large databases with many dimensions. Our method supports incomplete and heterogeneous objects (defined on different dimension subsets) and objects with large extensions on their dimensions.

Key-words: Approximation Model and Clustering for Multi-dimensional Objects

Unité de recherche INRIA Rocquencourt
Domaine de Voluceau, Rocquencourt, BP 105, 78153 Le Chesnay Cedex (France)
Téléphone : +33 1 39 63 55 11 — Télécopie : +33 1 39 63 53 30

RVA-clustering: Une Méthode d'Indexation d'Objets Multi-dimensionnels Basée sur un Modèle d'Approximation

Résumé : Dans ce papier nous proposons une nouvelle approche d'indexation pour répondre efficacement aux requêtes spatiales (intersections, inclusions, englobements) sur des larges collections d'objets multi-dimensionnels (hypercubes). De nombreuses applications peuvent bénéficier de notre technique comme par exemple la recherche et le traitement d'images, l'indexation de documents, les séries temporelles, les systèmes de notification, ainsi que d'autres applications comportant des données spatiales multi-dimensionnelles. Notre contribution consiste dans la définition d'un modèle d'approximation pour des objets multi-dimensionnels et opérations spatiales, qui accélère la vérification des objets et qui rend possible une organisation en clusters de la base de données, pour éviter une vérification exhaustive. La stratégie de groupement basée sur la probabilité d'accès permet à la clusterisation de bien se comporter pour des données et requêtes non uniformes. Une analyse de performances montre que notre approche gère efficacement de grandes collections d'objets multi-dimensionnels avec beaucoup de dimensions. Notre méthode supporte des objets incomplets et hétérogènes (définis sur différents sous-ensembles de dimensions) aussi bien que des objets dont certaines dimensions peuvent couvrir de larges intervalles.

Mots-clés : Modèle d'Approximation et Clusterisation pour Objets Multi-dimensionnels

1 Introduction

In this paper, we propose a new indexing approach for efficiently answering spatial queries over large collections of multi-dimensional objects.

1.1 Terms and Definitions

We start our presentation, by introducing the elementary notions used throughout the paper:

Set of Attributes: Let $\mathcal{A}$ be a *set of attributes*. Each *attribute* is associated with a continuous domain of values, given by its minimum and maximum value limits. Throughout the paper the terms *attribute* and *dimension* are used as synonyms.

Extended Objects: An *extended object* defines intervals for a subset of attributes of $\mathcal{A}$. The intervals are given by their minimum and maximum value limits, agreeing with the domains of values associated to the corresponding attributes. An extended object defining intervals for all the attributes of $\mathcal{A}$ is said to be *complete*. When an extended object defines only a subset of attributes of A, then we deal with an *incomplete* extended object.

Target Database: The *target database* is represented by a dynamic collection of extended objects. The extended objects from the target database will be next referred as *database objects*. Object insertions and deletions are subject to *update queries*.

Spatial Queries: The *spatial queries* serve to perform object selections over the target database. The selection criteria are based on *spatial operations*. Generally a spatial query specifies:

1. the type of the spatial operation: *intersection*, *containment*, or *enclosure*;
2. the extended object relative to which the spatial operation is performed: *query object*.

The database objects are required to satisfy the following two conditions to qualify for the query result:

1. *Attribute Compatibility Condition*: All the attributes of the database objects need to be supported by the query object.
2. *Spatial Operation Condition*: According to the type of the spatial operation, all the intervals of the database objects need to intersect, to be contained in, or to enclose, the corresponding intervals of the query object.

1.2 Problem Statement and Contributions

The most common multi-dimensional indexing technique dealing with spatial queries on extended objects is the R-tree technique. This method and related approaches (R^+-tree, R^*-tree, X-tree, SR-tree, SS-tree) rely on hierarchical organization of minimum bounding boxes. The minimum bounding boxes delimit minimal regions (hypercubes or hyperspheres) completely enclosing collections of database objects or sets of smaller bounding boxes. The

enclosure property, also known as the bounding property, has to be satisfied for all dimensions. Because of the enclosure property, these techniques are very sensitive to overlaps between bounding boxes. Region overlaps determine the exploration of multiple tree branches leading to serious performance degradation [9]. Overlaps occur for several reasons: First, the extended objects may overlap themselves. Second, incomplete extended objects lead to large overlaps since missing attributes have to be replaced by intervals covering full domains. Third, when the number of dimensions is large, the probability of overlap between bounding boxes is higher. This well known phenomenon is called the "curse of dimensionality" and explains the bad performance of the R-tree methods in high-dimensional spaces. The goal of our approach is to overcome these performance problems by proposing a new approach to cluster extended objects.

The originality of our clustering approach, called RVA-clustering, is that we do not use all dimensions for the object grouping, but only a subset of dimensions which are most selective for the qualifying objects. A second originality is that we use a "similarity" property instead of the enclosure property to cluster objects. For the RVA-clustering, two objects are considered similar if they define similar intervals on a subset of common attributes. Two intervals are similar if their min and max limits are close. With this notion of similarity, similar objects are likely to have similar probabilities of being accessed. Moreover, our cluster split strategy optimizes clustering by prioritizing the "small" intervals (or the intervals with low access probabilities) to be used as similar intervals. Another originality of our approach is that we use an *approximation model for extended objects* (in the spirit of VA-File for multi-dimensional points) to detect similar intervals and to speed up spatial comparison checks. Our method leads to a specific database organization which consists in:

1. *real database* represented by the collection of real extended objects
2. *approximated database* managing the object approximations corresponding to the real extended objects. Each object approximation has a reference to the real object in the real database.

The approximated database is clustered using our "similarity"-based strategy. A spatial query is executed as follows: First, the approximated database is checked to find objects approximations which match the query. RVA-clustering allows to avoid an exhaustive examination of the approximated database. The real objects corresponding to the approximated objects matching the query are then fetched from the real database. Since the check on approximation objects may not always give a sure answer some of the real objects may need to be re-checked on their real values. To optimize locality of real database accesses, we apply to the real database a clustering which follows the clustering of the approximated database. We assure a one-to-one mapping between clusters in the approximated database and clusters in the real database. This strategy allows us to optimize I/O costs when the real database is on disk.

The main advantage of our clustering based on subsets of dimensions is the lower probability of region overlap. For this reason, we can deal with databases containing large and/or

incomplete extended objects. Performance measurements also show that RVA-clustering supports high numbers of dimensions.

1.3 Applications

Our approach has a high potential for a wide range of applications. This includes domains related to image retrieval, document indexing, time series, interval-based notification systems, geographical systems, and other applications dealing with n-dimensional data and involving spatial operations. In particular, our model for data representation, based on approximation and supporting database filtering for spatial operations, can be used as a general technique to improve the access capabilities to high-dimensional region data.

Interval-based Notification System: As a direct application, we have integrated our solution in a prototype of a publish-subscribe notification system. In this event-driven system, the subscriptions define intervals for their attributes, and a high rate of intersection, containment and enclosure events, emitted by publishers, are verified against the subscription database. The role of the system is to retrieve and notify the subscribers matching the incoming events. An example of subscription is "Notify me of all new apartments with a rent price between 400$ and 1000$, having between 2 and 4 rooms, and located in the neighborhood of Newark". In this application, the subscriptions and the events can be represented as extended objects. Although the number of possible dimensions is usually large, the subscriptions generally define small subsets of attributes. As a consequence, the extended objects corresponding to subscriptions are usually incomplete and highly heterogeneous.

Clustering for Hyperspace Point Collections: We also tested our technique as clustering support for large collections of hyperspace points. The objective of this application is to accelerate the execution of point, range, and distance queries. For this purpose, the points are organized in clusters based on their spatial distribution. At physical level, the points from the same cluster are together stored in one or several pages of external support. With respect to the clustering dimensions, each cluster can be represented by an extended object corresponding to the bounding box that encloses the points stored at its level. The collection of extended objects is indexed in memory using our technique. When performing selections over a collection of points, the queries are first translated into spatial queries against the indexing structure. This step allows to quickly identify the clusters potentially containing candidate points. The qualifying pages are then read from the external support and the corresponding data points are checked using initial selection criteria. This type of application introduces particular object characteristics: (1) the extended objects define intervals for all the dimensions used to cluster the point collection; (2) the intervals are large for the regions in which the points are sparsely distributed, and small for the regions where the points are highly concentrated; (3) the extended objects do not overlap with each other and eventually cover the entire space.

The two applications above illustrate the need to cope with (i) incomplete and heterogeneous extended objects, (ii) large extended objects, and (iii) many possible dimensions.

1.4 Paper Organization

The rest of the paper is organized as follows: In Section 2 we present the related work. In Section 3 we present the approximation model for extended objects used to speed up the verification of the database objects and to support the grouping of the "similar" objects. In Section 4, we present the database organization according to our RVA-clustering method. In this context, we introduce the spatial query algorithm taking into account the database organization in clusters of object approximations. Section 5 discusses the database clustering and maintenance. It describes the cluster split strategy and presents algorithms for update operations like object insertions and deletions. Some implementation aspects related to database maintenance and dynamic restructuring are also considered. In Section 6, we validate the RVA-approach with a series of experimental results. Finally, conclusions and future work are presented in Section 7.

2 Related Work

The existing multi-dimensional indexing techniques are designed to manage either hyperspace points, or spatial objects with extensions. Two recent surveys [5, 9] review and compare most existing multi-dimensional access methods. There are two large families of solutions. The first family is based on the KD-tree approach, which consists in space partitioning alternating the split dimension: KDB-tree, hB-tree, Quad-tree, ϵ-tree. The second family originates from the R-tree structure and relies on hierarchical organization of bounding regions: R^+-tree, R^*-tree, X-tree, SR-tree, SS-tree. As a general rule, the KD-tree solutions are designed for collections of hyperspace points, while the R-tree methods can also manage objects with extensions (known as regions).

R-tree technique is the most common method that can deal with spatial queries over databases of extended objects. However, R-tree technique suffers from serious performance degradation when the number of dimensions is large. As explained in the introduction this degradation is mainly due to region overlap. Many techniques have been proposed to deal with this problem, especially trying to minimize the region overlap. Despite this effort, experiments reported in [3, 14] show that, for more than 5-10 dimensions, and considering just point data, the simple sequential scan, or filtering techniques based on approximated data representation and on sequential scan (VA-File in [14]), outperform complex R-tree implementations like R*-tree[1], X-tree[4], and Hilbert R-tree[8]. These results were obtained for collections of hyperspace points. In our case, the presence of objects with (possibly large) extensions determines wide overlapping regions, leading from the beginning to poor performance. Another important limitation of the R-tree methods is that the managed objects need to provide values for all dimensions to make possible the verification of the bounding property. In practice, some applications involve multi-dimensional objects defined on reduced subsets of attributes (e.g. attribute-based notification systems).

The Pyramid-Technique[3] is an original approach for indexing multi-dimensional points which relies on partitioning the space into pyramids and cutting these pyramids into slices.

This partitioning provides a mapping from a n-dimensional space to a 1-dimensional space alleviating the problem of the curse of dimensionality. However, such space partitioning can not be applied to extended objects.

Similarly to our technique, VA-File[13, 14], IQ-tree[2] and A-tree[12] use approximation models to optimize query performance. VA-File and IQ-tree rely on approximated representation to reduce the amount of data read during similarity searches, so accelerating the sequential scan. Such approach outperforms partitioning-based indexing techniques in high-dimensional spaces. The A-tree[12] is based on the R-tree structure and introduces Virtual Bounding Rectangles (VBRs) which contain and approximate minimum bounding rectangles (MBRs) and data objects. Comparing with SR-tree and VA-File, the authors of A-tree reported noticeable access time gains. However, all these techniques focus on multi-dimensional points and do not deal with extended objects.

In [7], the authors describe a publish-subscribe system where the subscriptions are organized in clusters based on access probabilities. More precisely, the subscriptions are clustered based on their most selective equality predicates (called access predicates). This technique cannot deal with inequality and interval predicates. By using an approximation model for intervals, our RVA-clustering supports inequality and interval predicates as access predicates.

In read-mostly environments, bitmap indexes can be also used for processing complex multi-dimensional queries [11]. A simple bitmap index associates a bit vector (bitmap) to each record that defines an indexed attribute. Commonly a bitmap has as many bits as distinct attribute values. According to the encoding schema (equality-, range- or interval-based) a number of bits are set to 1, allowing to quickly answer equality, range and membership queries [6]. The bitmap indexes are efficient for attributes with low cardinalities and discrete values. [11] proposes a bitmap index for attributes with real values, by partitioning the attribute domains in regions (bins) and assigning a bit to each distinct region. Again, these techniques only apply to multi-dimensional objects which can be seen as points in their attribute spaces. Extended objects are not supported.

3 Approximation Model for Extended Objects

In this section, we present the data approximation model serving as support for our database representation, organization and interrogation. This model provides: (i) an approximated representation for extended objects (subsection 3.1); (ii) approximated comparison criteria for common spatial operations like intersection, containment, or enclosure (subsection 3.2); and finally (iii) multi-level approximation support, to facilitate the grouping of "similar" objects (subsection 3.3).

3.1 Approximated Representation for Extended Objects

The approximated representation for extended objects relies on the partitioning of each attribute domain. The domains of values are divided into distinct regions called *domain*

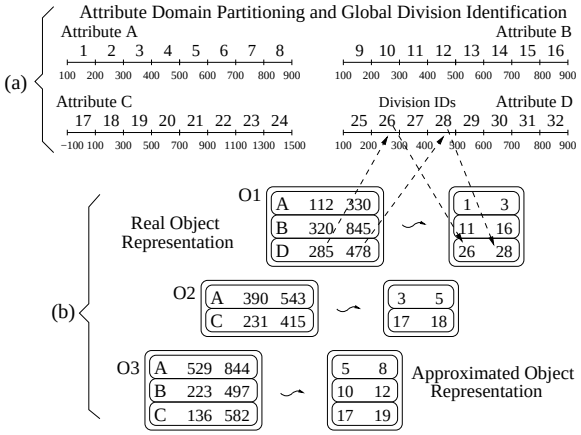


Figure 1: Representation for Extended Objects

divisions. Passing from one attribute to another, numerical identifiers are sequentially assigned to domain divisions. With this partitioning, the *approximated representation of an interval on a given attribute* is provided by the pair of identifiers associated with the two domain divisions which respectively contain the minimal value and maximal value of the interval. The *approximated representation of an extended object* consists of the collection of approximated representations of its intervals.

Example 1 : Figure 1 shows examples of approximated representations. The attributes A, B, and D have domain of values ranging between 100 and 900, while the attribute C ranges between -100 and 1500. Figure 1(a) depicts the attribute domain partitioning and the global division identification: each domain of values is divided into eight equally-sized regions; all the regions are associated with numerical identifiers from 1 to 32. Based on this partitioning schema, Figure 1(b) shows three database objects, together with their approximated representations.

3.2 Approximated Spatial Comparison

The *approximated spatial comparison* computes an approximated result of a spatial comparison (intersection, containment, or enclosure) between two extended objects. The result of the approximated spatial comparison takes one of the following values: (i) *yes*, when the spatial operation is verified, (ii) *no*, when the spatial operation is not verified and (iii) *perhaps*, when the result of the spatial operation is uncertain. The approximated spatial comparison individually applies to the common attributes of the two compared objects. In the next subsection, we explain the one-attribute approximated comparison, then we present the multi-attribute spatial comparison.

3.2.1 One-Attribute Approximated Comparison

Let A be an attribute, and let I_1 and I_2 be two intervals on A. The approximated spatial comparison of I_2 with respect to I_1 is performed in two steps:

Step 1: Domain divisions annotation. First, we annotate the domain divisions of attribute A with respect to the first interval I_1. The domain divisions can be in one of the following situations:

- located in the left side of I_1 (to express this case we use the symbol l);
- overlapping the left edge of I_1 (li);
- located inside I_1 (i);
- overlapping the right edge of I_1 (ir);
- located in the right side of I_1 (r);
- completely enclosing the interval I_1 (e).

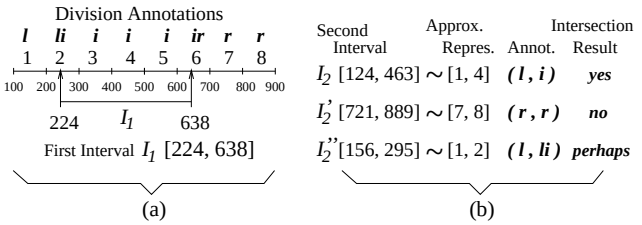


Figure 2: One-Attribute Approximated Comparison

An example of domain division annotation is given in Figure 2(a). In this example, the domain divisions of the attribute A (preserved from Figure 1) are annotated with respect to the interval I_1. For instance, the first domain division has the label l due to its location in the left side of I_1. The division 4 is annotated with the symbol i because it is entirely included in I_1. The division 7 has the label r reflecting its position in the right side of I_1.

Step2: Derivation of spatial comparison results. The second step uses as input the domain division annotations computed in the first step and selects the annotation pair which corresponds to the approximated representation of the second interval I_2. The result of the spatial comparison is decided based on this annotation pair. Figure 2(b) illustrates the approximated intersection comparison involving the first interval, I_1, and several possible second intervals, I_2, I_2', and I_2''. The intersection is decided based on the pairs of annotations corresponding to the second intervals. For example, let us first consider the pair (l, i) associated with the second interval I_2. The two annotations have the following meaning: the interval I_2 begins in a region located in the left side of the interval I_1 and ends in a region completely enclosed by I_1. Obviously, the intervals I_2 and I_1 overlap each other, so

the result of the intersection test is *yes*. Now, let us examine the pair of annotations (r, r) associated with the second interval I_2'. According to our notation, this interval begins and ends in two regions both located in the right side of the interval I_1. As a consequence, the two intervals are completely disjoint and the result of the intersection test is *no*. Another interesting case corresponds to the pair (l, li) associated with the second interval I_2''. I_2'' begins in a region located in the left side of I_1 and ends in a region that overlaps the left edge of I_1. In this case, the two intervals might or might not overlap. The intersection is uncertain and the answer is *perhaps*. The final decision can be taken only by examining the real limits of the two compared intervals. The results of approximated spatial comparisons for all possible annotation pairs are given in the spatial decision tables from Figure 3. These decision tables give the comparison results for each of the three common spatial operations: intersection, containment, and enclosure. In these tables, *y* stands for *yes*, *n* for *no*, and *p* for *perhaps*. The combinations of annotations that cannot occur in practice are marked with the minus symbol.

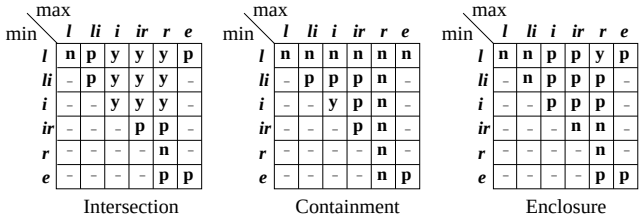


Figure 3: Annotation-based Comparison Results

3.2.2 Multi-Attribute Approximated Comparison

Based on the individual attribute comparisons, the overall result of the approximated spatial comparison is simply computed as follows: if at least one of the individual results is *no*, then the overall comparison result is *no*; otherwise, if at least one of the individual results is *perhaps*, then the overall result is *perhaps*; finally, if all the individual comparison results are *yes*, the overall result is *yes*.

3.2.3 Comparing a query object to a set of database objects

The reader may be surprised by the asymmetry of our comparison procedure. This asymmetry is due to the fact that, in practice, we want to compare one object (the query object) with a set of objects (the database objects). Using our asymmetrical procedure we compute only once the annotations of all attribute domain divisions with respect to the query object, and iterate over the database objects to perform the second step of the comparison.

3.3 Multi-Level Domain Partitioning

The approximated object representation looses in accuracy compared to the real object representation. As a consequence, distinct real intervals have identical approximated representations and different extended objects share (entirely or partially) their approximations. We actually rely on this property to group extended objects into clusters. To better support the object grouping, we extended the initial partitioning model to a multi-level domain partitioning. Such partitioning, illustrated in Figure 4, involves multiple levels of domain divisions and allows successive interval approximations. The multi-level partitioning model

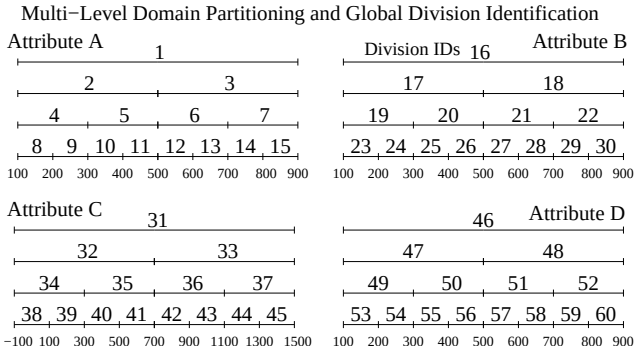


Figure 4: Heap-like Domain Partitioning

is determined by two parameters: (1) the number of levels of recursive divisions; (2) the division factor, representing the number of divisions per region. In Figure 4, we have 3 partitioning levels, and a division factor of 2 divisions per region. The domain divisions from all the levels are associated with numerical identifiers allocated in a heap-like fashion. Such global division identification enables a serialized representation for the partitioning schema. This schema facilitates the task of division annotation which is necessary when performing approximated spatial comparisons.

Note that the approximated spatial comparisons now involve object approximations defined with respect to the multi-level domain partitioning model.

3.3.1 Interval Approximations and Coverage Relationship

Because of the multi-level domain partitioning, multiple approximated representations can be defined for the same interval, based on different partitioning levels. The higher partitioning levels are less accurate than the lower ones. As a consequence, a *coverage relationship* can be established between different approximations of the same interval. Accordingly, an interval approximation defined at a higher partitioning level implicitly covers all the interval approximations defined for the same interval at lower partitioning levels (including itself and the real interval).

Example 2 : Considering the multi-level domain partitioning of the attribute A from Figure 4, the successive approximations of the real interval $I_1 = [230, 485]$ are $\{1, 1\}$, $\{2, 2\}$, $\{4, 5\}$ and $\{9, 11\}$. The coverage relationship between these interval approximations is dictated by the domain partitioning levels used to define them. For instance, the interval approximation $\{2, 2\}$ covers the lower level approximations $\{4, 5\}$ and $\{9, 11\}$.

3.3.2 Object Approximations and Coverage Relationship

The approximated representation of an extended object is given by the collection of pairs of division identifiers corresponding to the interval approximations associated with the object attributes. Due to the multi-level domain partitioning, the interval approximations corresponding to distinct attributes can be defined on different domain partitioning levels. As a consequence, the same extended object can have multiple approximated representations. The coverage relationship between different object approximations is based on the coverage notion defined for interval approximations. An object approximation ω_1 covers another object approximation ω_2 if and only if for each attribute represented in ω_1 there exists the same attribute in ω_2, and the interval approximation defined for this attribute in ω_1 covers the corresponding interval approximation from ω_2.

Example 3 : Again considering the multi-level domain partitioning from Figure 4, the real extended object

$$O =< A[112, 330],\ B[320, 845],\ D[285, 478] >$$

has several possible approximated representations like

$$\omega_1 = < A\{08, 10\},\ B\{20, 22\},\ D\{49, 50\} >$$
$$\omega_2 = < A\{02, 02\},\ B\{17, 18\},\ D\{47, 47\} >$$
$$\omega_3 = < A\{08, 10\},\ B\{25, 30\},\ D\{54, 56\} >.$$

The coverage relations between these three object approximations are: ω_1 *covers* ω_3, ω_2 *covers* ω_1, and ω_2 *covers* ω_3.

3.3.3 Related Definitions and Properties

Complete and Incomplete Object Approximations: An object approximation is said *complete* if all the attributes of the real object are represented in the object approximation. Otherwise, we deal with an *incomplete object approximation.*

Example 4 : The object approximations ω_1, ω_2, and ω_3, from Example 3, are all complete. An example of incomplete approximation of

$$O = < A[112, 330],\ B[320, 845],\ D[285, 478] >$$

is $\omega_4 =< A\{02, 02\},\ B\{17, 18\} >$.

Note that, according to our notion of coverage, a first object approximation, ω_a, covering a second object approximation, ω_b, can be incomplete relative to the last one. In other words, the coverage relationship allows the second object approximation to contain attributes without any corresponding covering intervals in the first object approximation. For instance, the incomplete object approximation ω_4 from Example 4 covers the complete object approximations ω_1, ω_2, and ω_3 from Example 3.

Fine and Coarse Interval Approximations: An interval approximation based on the last level of domain partitioning is considered a *fine interval approximation*. An interval approximation defined on a higher partitioning level is considered a *coarse interval approximation*.

Fine and Coarse Object Approximations: An object approximation is considered *fine* if all the interval approximations associated with its attributes are fine. Otherwise, we deal with a *coarse object approximation*. In Example 3, ω_3 is a fine object approximation, while ω_1 and ω_2 are coarse object approximations.

4 Database Organization and Spatial Query Algorithm

The general organization of the target database relies on the approximation model for extended objects described in the previous section. Based on this model, we introduce now our proposition for data representation and organization, and the corresponding spatial query algorithm.

4.1 Data Representation and Organization

Data Representation The extended objects from the target database are stored using both the real and the approximated object representations. As a consequence, the target database consists of: (i) *real database*, represented by the collection of real extended objects, and (ii) the *approximated database*, managing the object approximations corresponding to the real objects. With respect to the multi-level domain partitioning model, the object approximations associated to real database objects are complete and fine object approximations. The one-to-one association between real objects and object approximations, is assured through references based on object identifiers.

Data Organization The object approximations from the approximated database are grouped in clusters represented by *cluster signatures*. The cluster signatures are object approximations defined on higher domain partitioning levels and covering the object approximations represented at the cluster level. For this reason, the cluster signatures are usually coarse and incomplete object approximations.

Example 5 : Based on the schema from Figure 4, let us consider an initial database containing the following database objects

$$O_1 =< A[221, 893],\ B[514, 867],\ C[1120, 1435],\ D[114, 803] >$$
$$O_2 =< B[689, 741],\ C[930, 1080],\ D[443, 822] >$$
$$O_3 =< A[112, 245],\ B[503, 810],\ C[1205, 1294],\ D[602, 636] >$$
$$O_4 =< A[289, 301],\ D[512, 536] >$$

and their fine and complete approximated representations

$$\omega_1 = < A\{09, 15\},\ B\{27, 30\},\ C\{44, 45\},\ D\{53, 60\} >$$
$$\omega_2 = < B\{28, 29\},\ C\{44, 44\},\ D\{56, 60\} >$$
$$\omega_3 = < A\{08, 09\},\ B\{27, 30\},\ D\{58, 58\} >$$
$$\omega_4 = < A\{09, 09\},\ D\{57, 57\} > .$$

We want to cluster these objects into two groups. There are many possible clustering solutions. A good clustering is a clustering where the intervals approximated by the cluster signatures are as small as possible so that the probability of access is as low as possible. In the considered example, a good clustering solution would be to group the object approximations ω_1 and ω_2 in a cluster represented by the signature $\sigma_{12} = < B\{18, 18\},\ C\{37, 37\} >$ based on the attributes B and C, and to place together ω_3 and ω_4 in another cluster represented by the signature $\sigma_{34} = < A\{04, 04\},\ D\{51, 51\} >$ which corresponds to the attributes A and D. Note that σ_{12} covers the object approximations ω_1 and ω_2, while σ_{34} covers the object approximations ω_3 and ω_4. To construct these clusters we have exploited the fact that both O_1 and O_2 have smaller "similar" intervals on attributes B and C and that O_3 and O_4 have smaller "similar" intervals on attributes A and D. By grouping objects according to these "tiny" signatures we guarantee that the clusters would be accessed only if the spatial query intersects the corresponding small intervals. Let us note that such clustering would have been impossible using a R-tree approach. Indeed, since object O_3 is included in object O_1 a clustering based on containment would force to put O_3 and O_1 in the same cluster.

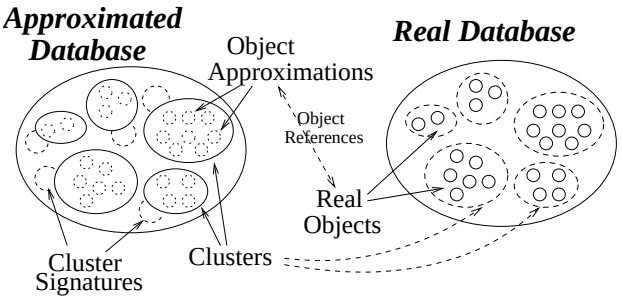


Figure 5: Database Organization Model

Figure 5 illustrates our database organization model. The real database follows the clustering organization of the approximated database.

4.2 Spatial Query Algorithm

The approximated object representation and the organization in clusters of the approximated database are meant to improve the spatial query performance. When answering spatial queries, the verification of the database objects is primarily supported by approximated spatial comparisons between query objects and database object approximations. The organization in clusters of the approximated database allows to avoid an exhaustive object verification. Next we present the corresponding spatial query algorithm:

Spatial Query Algorithm **{**
Notations:
$\mathcal{RD}$ - real database;
$\mathcal{AD}$ - approximated database;
Ψ - real spatial comparison (intersection,
containment, or enclosure)
ψ - approximated spatial comparison
Input: ρ - query object;
Output: $\mathcal{R} \subseteq \mathcal{RD}$ - result set containing the objects
selected by the spatial query;
Variables:
$\mathcal{V}$ - set of objects which need to be verified using
the real spatial operation Ψ;
Initialization:
$\mathcal{R} \leftarrow \emptyset$; $\mathcal{V} \leftarrow \emptyset$;
Phase 1. *Approximated Spatial Comparison:*
Step 1.0 *Division annotation wrt the query object* ρ*;*
as explained in section 3.2.1
Step 1.1 *Cluster Selection:*
for each cluster signature σ from $\mathcal{AD}$ **do {**
if $(\psi(\sigma, \rho) \in \{yes, perhaps\})$ **then {**
Step 1.2 *Object Selection:*
let c **be** the cluster represented by σ;
for each object approximation $\alpha \in c$ **do {**
if $(\psi(\alpha, \rho) = yes)$ **then** $\mathcal{R} \leftarrow \mathcal{R} \cup \delta$ **else**
if $(\psi(\alpha, \rho) = perhaps)$ **then** $\mathcal{V} \leftarrow \mathcal{V} \cup \delta$;
// where $\delta \in \mathcal{DR}$ is the database object
// represented by α.
}}}
Phase 2. *Real Spatial Comparison:*
for each database object $\delta \in \mathcal{V}$ **do {**
if $(\Psi(\delta, \rho) = yes)$ **then** $\mathcal{R} \leftarrow \mathcal{R} \cup \delta$;
} // **for**
} // ***Spatial Query Algorithm***.

In *Phase 1*, approximated spatial comparisons are used to support:

1. the selection of the clusters potentially containing candidate objects for the query answer, by examining the associated cluster signatures (*Step 1.1*);

2. the selection of the database objects qualifying for the query answer, by verifying the object approximations stored at the cluster level (*Step 1.2*).

The database objects for which the approximated comparison results are *yes* are directly added to the query answer set $\mathcal{R}$. The database objects for which the comparison results are *perhaps* are placed in the auxiliary set $\mathcal{V}$ because they need to be verified using real spatial comparison criteria. The verification using real spatial comparison criteria is performed in *Phase 2* of the algorithm. This phase completes the result set $\mathcal{R}$ with the qualifying database objects from $\mathcal{V}$.

Considerations: The database organization in clusters allows us to avoid the exhaustive object examination. Only the objects from the clusters whose associated signatures satisfy the approximated comparison criteria need to be examined. Another advantage of using approximated comparison is the fact that approximated spatial comparisons are faster than the real spatial comparisons. This happens because the domain annotation relative to the query object is performed only once per spatial query and does not depend on the number of database objects (see subsection 3.2). Different types of spatial operations (intersection, containment, or enclosure) can be managed by simply switching the table used to look up the comparison decision results (see Figure 3).

Note that the cluster signatures and the database object approximations are verified using the same approximated spatial comparison criteria (ψ in the algorithm). If the number of clusters is very large, the cost of scanning and checking all the signatures may become important. A solution to reduce this cost would be to cluster the signatures using the same clustering approach as we used for database objects. This is clearly possible since signatures are defined as (approximated) extended objects. This method can be generalized leading to a multi-level clustering. In this paper, we focus on the validation of the simple (one-level clustering) approach and do not explore such enhancement. Experiments show that the one-level clustering is sufficient even for relatively large collections of objects (2 millions of extended objects).

5 Database Clustering and Maintenance

The database organization relies on the partitioning in clusters of the approximated database. In this section, we focus on the clustering strategy, and on aspects related to database maintenance (object insertions, deletions, and cluster restructuring). In general, a database object can be placed in any of the clusters whose signatures cover the given object. Our goal is to place each database object in the cluster that minimizes the object probability of being checked.

5.1 Database Clustering

For the database clustering, we propose an incremental approach based on successive cluster splits. We start with one single cluster storing the object approximations associated with the initial set of extended objects. When this cluster becomes too large, new clusters are created partitioning the existing objects. The initial cluster, next referred as the *root cluster*, has no associated signature and can accommodate any object. For this reason, the root cluster will permanently serve to manage objects not supported by any other clusters.

Cluster Split Decision When performing spatial queries, during cluster explorations, all the object approximations stored at the cluster level are verified using approximated spatial comparison criteria. If too many objects are represented in the same cluster, the verification cost becomes important. However, the general performance is not seriously affected if only a few queries trigger the cluster exploration. The real problem raises when such large cluster is frequently explored. In such situations, the general answering time increases with a term proportional to the fraction of objects managed at the cluster level. When in average a large proportion of these objects matches the approximated spatial criteria, then the frequent cluster exploration is determined by the general query selectivity and the verification cost can not be avoided. On the contrary, when many objects at the cluster level do not satisfy the selection criteria, then a lot of useless checks are performed, leading to a deficit in the cluster checking performance. To alleviate this problem, the large cluster can be split in several smaller clusters with lower probabilities of being accessed by the same queries. Indeed, if the new clusters are explored alternatively, important time gains can be obtained on average. For these reasons, a cluster split is decided when the following three conditions are at once fulfilled:

1. the probability of accessing the cluster is over a given threshold (noted max_access)
2. the cluster size exceeds a given size threshold (noted max_size).
3. the average percentage of successful checks is under a given threshold (noted $min\_success$)

Such situation can be easily detected based on statistics on the incoming queries, associated with the existing clusters.

Sub-Cluster Creation The splitting procedure consists in dispatching the objects from the parent cluster into sub-clusters whose signatures are covered by the signature of the parent cluster. The number of possible covered signatures is very large. For this reason, we limit the possible sub-cluster signatures by considering only signatures which are *simple extensions* of the parent cluster signature. A simple extension of a parent signature w_p is constructed either by (i) replacing an approximated interval I_p of ω_p by a more precise interval defined on a domain sub-division of I_p or by (ii) adding an approximated interval defined on an attribute which is not referenced in the parent signature w_p.

Example 6 : Let $\omega_p =< A\{02,02\},\ B\{18,18\} >$ be the signature of a cluster. The set of simple extensions of ω_p consists of the following set of object approximations:

$$\begin{array}{l}
\{ < A\{02,02\},\ B\{18,18\},\ D\{47,47\} > , \\
\ \ < A\{02,02\},\ B\{18,18\},\ D\{47,48\} > , \\
\ \ < A\{02,02\},\ B\{18,18\},\ D\{48,48\} > , \\
\ \ < A\{02,02\},\ B\{18,18\},\ C\{32,32\} > , \\
\ \ < A\{02,02\},\ B\{18,18\},\ C\{32,33\} > , \\
\ \ < A\{02,02\},\ B\{18,18\},\ C\{33,33\} > , \\
\ \ < A\{04,05\},\ B\{18,18\} > \ ,\ < A\{04,04\},\ B\{18,18\} > , \\
\ \ < A\{05,05\},\ B\{18,18\} > \ ,\ < A\{02,02\},\ B\{21,21\} > , \\
\ \ < A\{02,02\},\ B\{22,22\} > \ ,\ < A\{02,02\},\ B\{21,22\} > \}.
\end{array}$$

Among the possible sub-clusters, we choose the ones containing enough objects (above a threshold noted min_size) and having the signatures with the lowest probabilities of access. Local statistics about access probabilities of each simple extension of the cluster signature are maintained along the life of the cluster to support the split decision.

Cluster Split Procedure When splitting a cluster we need to identify the best cluster candidates for the object partitioning and to construct the signatures associated with the new clusters. The cluster split procedure is presented next:

Cluster Split Procedure {
Step 1. Find the simple extension of the current cluster signature, having the lowest access probability and covering enough object approximations for a new cluster (above min_size); if no such signature can be found, then go to ***Step 5***;
Step 2. Create a new cluster and set as its signature the "simple extension"-signature selected in ***Step 1***;
Step 3. Transfer to the new cluster all the object approximations from the original cluster, which are covered by the new cluster signature;
Step 4. If too many objects approximations still remain in the original cluster, then repeat from ***Step 1***;
Step 5. If too few objects approximations remain in the original cluster, then reinsert them in the approximated database and completely remove the original cluster.
} // ***Cluster Split Procedure***

As it can be noticed, not all the object approximations from the original cluster are necessarily moved to the newly-created clusters. Actually, the split procedure ends when enough

object approximations are removed from the original cluster. If very few object approximations remain in the original cluster, the cluster is removed and its objects are reinserted in the approximated database using the procedure described in subsection 5.2 bellow. Our split strategy has several advantages: First, the database objects are grouped based on their less accessed attributes (e.g. with small interval extensions). Very large objects (with large intervals on all dimensions, and thus frequently accessed) remain close to the root cluster. Second, incomplete and heterogeneous database objects are well managed being clustered based on their subset of attributes. These properties explain why our clustering method is not affected by large, incomplete, or heterogeneous objects.

5.2 Object Insertions

When inserting a new object in the target database, we aim to place it in a cluster that minimizes the object probability of being checked without qualifying for the query result. For this purpose, we need to identify the clusters able to accommodate the new object, and to choose the one that best suits our placement strategy. The object insertion procedure is presented next:

Object Insertion Procedure {
Step 1. Annotate the domain divisions corresponding
to the attributes defined by the new object relative
to the intervals specified for these attributes;
Step 2. Verify the signatures of the existing clusters
and identify the clusters that can accommodate the
new object (their signatures cover the new object);
Step 3. Select the cluster with the smallest probability
of being accessed, and place the new object at its level.
} // ***Object Insertion Procedure***

The verification of the cluster signatures is performed using an approximated spatial comparison based on the decision table from Figure 6. This corresponds to a spatial operation that assures the coverage relation between the cluster signature and the new inserted object. When several candidate clusters can accommodate the new object, we simply select the cluster with the smallest probability of being accessed. The access probability can be dynamically maintained based on access statistics associated with clusters. However, if such statistics are not available, we simply prioritize the clusters whose signatures have the less extended interval approximations.

Note that when none of the non-root clusters can accommodate the new object, this last is assimilated by the root cluster, which implicitly has the highest access probability (equal to 1, because all queries explore this cluster).

min \ max	*l*	*li*	*i*	*ir*	*r*	*e*
l	**n**	**n**	**n**	**n**	**n**	**n**
li	–	**n**	**n**	**y**	**n**	–
i	–	–	**n**	**n**	**n**	–
ir	–	–	–	**n**	**n**	–
r	–	–	–	–	**n**	–
e	–	–	–	–	**n**	**y**

Figure 6: Coverage Decision Table

5.3 Object Deletions

The object deletion is performed based on the object identifier. The real object is removed from the real database, and the approximated database is updated accordingly. After several object deletions, some clusters can become too small. In practice it is not convenient to maintain clusters that contain very few objects. Our solution is to eliminate the almost-empty clusters by reinserting the remaining object approximations in the approximated database.

5.4 Cluster Restructuring

When the distribution of the selection queries changes over time, some of the database clusters can become very frequently accessed. Such situations can be detected based on access and selectivity statistics associated with clusters. When many of the database objects represented at the cluster level qualify for the query result the frequent access is due to the query selectivity. The problem appears when the cluster is frequently explored, but very few objects are finally selected. This indicates that the approximated intervals from the cluster signature are no more discriminatory relative to the objects represented at the cluster level. Such situations can be dealt with by removing the concerned clusters and reinserting the contained objects in the approximated database.

6 Experimental Evaluation

To verify the efficiency of our indexing solution, we performed extensive experimental evaluations over synthetic and real data sets with many dimensions (up to 40). We compare our RVA-performance to the sequential scan. Remark that it has been shown in [3, 14, 10] that R-tree like techniques are outperformed by the sequential scan for range queries over multi-dimensional point databases with more tha 5-10 dimensions. Thus, by comparing our RVA-technique to the scan we implicitly compare RVA-technique to the family of R-tree techniques. In our experiments RVA-clustering shows a speed-up factor up to 27 for uniform data, and up to 79 for real data sets with 10 dimensions. Moreover, with very large

number of dimensions (40 dimensions) RVA-clustering is between 6 times (uniform data) and 18 times (skewed data) faster than the sequential scan.

6.1 Experimental Setup

We ran all experiments on a Pentium III workstation with an i686 CPU at 650MHz and 768MB RAM operating under Linux.

Database Representation Parameters: The extended objects used in our tests were generated either synthetically, following an uniform distribution, or based on a real data set. In all cases, the domains of values of all dimensions were normalized to $[0, 1]$. For the approximated object representation and for the database clustering, we implemented the multi-level domain partitioning described in Section 3.3. For each dimension, we considered 4 levels of domain partitioning, with a division factor of 4. As a result, the domain partitioning schema of each dimension consisted of 340 divisions, from which 256 divisions on the last partitioning level.

Database Clustering Parameters: For the incremental organization in clusters of the target database (described in Section 5.1), we had to set the following clustering parameters:

- max_size – minimal size of a cluster considered for a split: 1000 object approximations;
- max_access – minimal access probability of a cluster considered for a split: 2.5%;
- $min\_success$ – maximal percentage of successful checks allowed for a cluster considered for a split: 10%;
- min_size – minimal size of a sub-cluster to be created, or of an existing cluster not to be removed: 100 object approximations.

We empirically determined and experimentally used the values above. Although the clustering parameters could have been better tuned from case to case (e.g. based on the data space dimensionality), we used the same values in all experiments, to better compare and evaluate the general performance of the RVA-approach.

Execution Parameters: In our tests, we varied the following execution parameters:

- *number of extended objects* in the target database: between 250,000 and 2,000,000 database objects;
- *number of dimensions* defined by the database/query objects: from 8 to 40 dimensions;
- *query selectivity* (number of database objects qualifying for the query result versus total number of database objects): we controlled the query selectivity by varying the average sizes of the intervals specified for the database/query objects. In our experiments, the query selectivities were between 1/10 and $1/1,000,000$ selected objects[1].

[1] A query selectivity of x/y, where $y \geq x$, means that x of y objects are in average selected for the query result.

Performance Parameters: In each experiment, a large number of spatial queries (10,000) were addressed to the indexing structure in order to evaluate the following performance parameters:

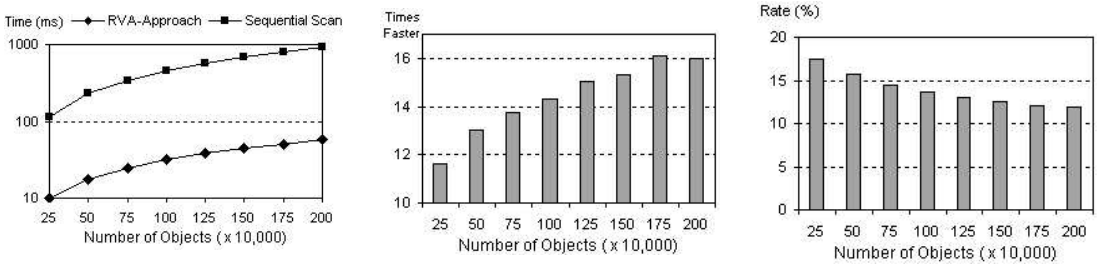
- *query answering time* and *speed-up factor* compared to sequential scan; These parameters give the performance in memory of our technique, assuming that the target database is managed in main memory.
- *approximated verification rate* (number of object approximations verified using approximated spatial comparisons, versus total number of object approximations); This parameter allows to estimate the order of the I/O cost, assuming that the approximated database is managed on external support.
- *real verification rate* (number of database objects verified using real spatial comparisons, versus total number of database objects); This parameter allows to evaluate the filtering efficiency of the approximation model. Remember that only the database objects for which the results of the approximated spatial comparisons are uncertain need to be verified against real spatial comparison criteria.

6.2 Experiments using synthetic data

Our synthetic data consisted of uniformly distributed hypercubes in a data space containing up to 40 dimensions.

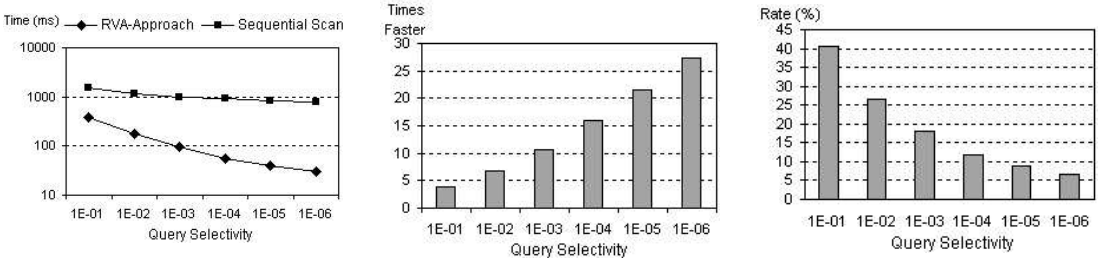
Varying number of objects (Figure 7) In the first experiment, we evaluated the performance behavior of RVA-approach with varying number of objects. For this purpose, we considered databases containing from $250,000$ to $2,000,000$ extended objects in a 16-dimensional space and performed intersection queries with $1/10,000$ selectivity. To assure the required selectivity, we generated database objects with interval sizes uniformly distributed between 0.0 and 1.0 with an average value of 0.333, and query objects with interval sizes between 0.0 and 0.44 with an average value of 0.2. Figure 7(a) shows the evolution of the query answering time for RVA-approach and sequential scan[2]. Considering first the sequential scan, the average answering time increases linearly with the number of database objects, from 115ms to 915ms. This evolution is normal, since all the database objects are compared to the query object. Regarding the RVA-approach, the average answering time evolves from 10ms to 57ms, proving out speed-up factors of 12 to 16 compared to the sequential scan (Figure 7(b)). The performance of the RVA-approach is explained by: (i) the database clustering which spares lots of useless object verifications; and (ii) the approximated spatial comparison which is between 2 and 3 times faster than the real spatial comparison. Relative to the first aspect, the approximated verification rate decreases from 17.54% to 11.87%, as illustrated in Figure 7(c). Actually, the approximated verification rate is determined by the clustering quality. In this experiment, the quality of the clustering

[2]Please note the logarithmic scale.



(a) Query Answering Time (b) RVA vs. Scan Speed-Up Factor (c) Approximated Verification Rate

Figure 7: Uniform Workload: Varying number of objects



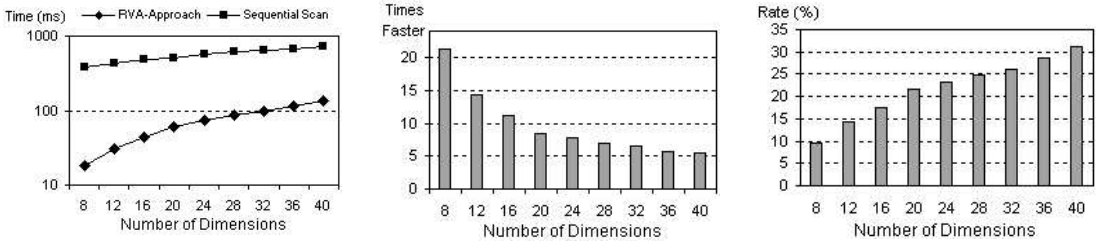
(a) Query Answering Time (b) RVA vs. Scan Speed-Up Factor (c) Approximated Verification Rate

Figure 8: Uniform Workload: Varying query selectivity

increases with the number of objects, because of the higher probability of finding objects with "similar" intervals on the same dimension subsets.

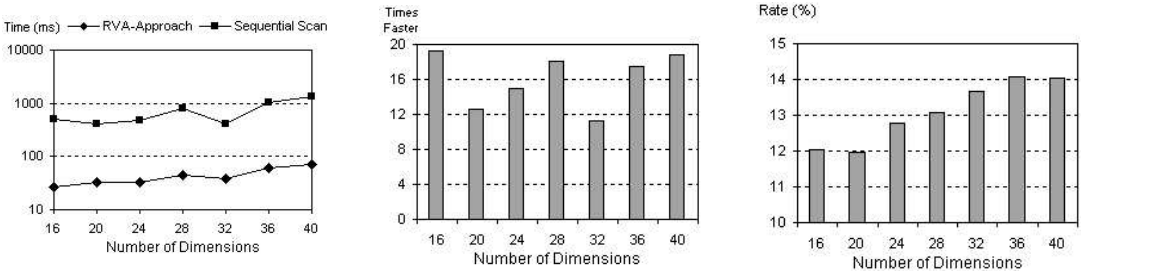
When the approximated database is managed on external support, the approximated verification rate gives the I/O cost. Because of the approximation model, the size of the approximated database is in average two times smaller than the size of the real database. This property allows to divide the I/O cost by two. More precisely, in this experiment, between 9% and 6% of the initial data size needs to be accessed from the external support to answer a spatial query. In the same context, another important parameter is the real verification rate which gives the percentage of database objects for which the results of the spatial comparisons cannot be decided based on the approximation model. The real verification rate proved to be very small in all our tests (between 0.0016% and 0.00175%). This demonstrates the efficiency of our approximation model for extended objects and spatial operations.

Varying query selectivity (Figure 8) The second experiment shows the impact of the query selectivity on the answering performance. For this test, we considered 2,000,000 extended objects in a 16-dimensional data space and evaluated the performance behavior for intersection queries with the following selectivities: $1/10$, $1/100$, $1/1,000$, $1/10,000$



(a) Query Answering Time (b) RVA vs. Scan Speed-Up Factor (c) Approximated Verification Rate

Figure 9: Uniform Workload: varying number of dimensions



(a) Query Answering Time (b) RVA vs. Scan Speed-Up Factor (c) Approximated Verification Rate

Figure 10: Skewed Workload: varying number of dimensions

and $1/1,000,000$. The experimental results are depicted in Figure 8. Considering first the sequential scan, the query answering time decreases slightly with the query selectivity, from 1548ms to 804ms (Figure 8(a)). This evolution is explained by the fact that when the query selectivity is higher, the probability of intersection with the query object is lower, and fewer dimensions need to be verified to reject a non-qualifying object. A similar phenomenon affects the RVA-approach, but the rapid dropping of the answering time for higher query selectivities is mostly due to the database clustering, which avoids lots of useless verifications. The approximated verification rate is depicted in Figure 8(c), and the speed-up factor compared to sequential scan is illustrated in Figure 8(b). Note that for the lowest query selectivity (1/10), where more than 40% of the object approximations are verified, the speed-up factor compared to the sequential scan is of 4. In contrast, for the highest selectivity (1/1,000,000) only 6% of object approximations are verified and the speed-up factor is of 27.

Varying number of dimensions (Figure 9) In the third experiment, we varied the number of dimensions to determine the influence of the space dimensionality on the query performance. For this purpose, we respectively considered 1,000,000 extended objects defined in data spaces with 8, 12, 16, 20, 24, 28, 32, 36 and 40 dimensions. The database objects were uniformly generated in $[0,1]^d$ (where d represents the space dimensionality),

leading to an average interval size of 0.333 per dimension. For each dimensionality, we performed 10,000 intersection queries with a constant selectivity of $5/10,000$. This selectivity was assured through the sizes of the intervals of the query objects. For example, in the 8-dimensional data space, the sizes of the query intervals were between 0.0 and 0.084 with an average value of 0.041. In the 40-dimensional data space, the sizes of the query intervals were between 0.279 and 1.0 with an average value of 0.518. As illustrated in Figure 9, the performance of the RVA-Approach decreases with the number of dimensions from 18ms in the 8-dimensional space to 136ms in the 40-dimensional space. This evolution is determined by the approximated verification rate which increases from 10% to 31% (Figure 9(c)). The large spatial extension of the query object in the 40-dimensional space (more than 0.5 in average for each dimension) explains the high verification rate and the reduced performance behavior of the RVA-approach. According to Figure 9(b), in the 40-dimensional space, the RVA-approach is only 6 times faster than the sequential scan, compared to 22 times faster in the 8-dimensional space. The main cause of the performance deterioration is the uniform distribution of the interval sizes of the database/query objects. Although our performance is always better than the sequential scan, such data/query spatial distribution is neither favorable for our technique, nor common in practice. The next experiment shows the performance behavior of the RVA-approach in a skewed scenario where the database objects have different interval size distributions on distinct dimensions.

Varying number of dimensions for skewed data (Figure 10) In the forth experiment, we assume that some of the intervals defined by the database objects are rather small, while the rest of the intervals are rather large. To respect this assumption, we generate database objects such that the probability of intersection of 75% of their dimensions (called *broad* dimensions), with the corresponding dimensions of the query object, is two times higher than the probability of intersection of the rest of 25% of dimensions (called *tight* dimensions). As a result, the broad dimensions specify large intervals, while the tight dimensions small intervals. Considering for instance a 40-dimensional extended object, 30 of its dimensions have interval sizes between 0.777 and 1.0 with an average value of 0.851 (which is considerably large), and the rest of 10 dimensions have interval sizes between 0.0 and 0.276 with an average value of 0.13 (rather small). An important observation is that not the same dimensions are tight/broad for all database objects. Actually, the 25% tight dimensions are randomly assigned for each database object. In our tests, we considered data spaces with 16, 20, 24, 28, 32, 36 and 40 dimensions, and assured each time a constant query selectivity of $5/10,000$. Comparing to the previous experiment, where we also varied the space dimensionality, in this case, the constant query selectivity was obtained through the interval sizes of the database objects, also taking care to respect the considered scenario. The query objects were uniformly generated in the data space $[0,1]^d$. The results of our tests are illustrated in Figure 10. This time, the RVA-Approach clearly takes advantage of the non-uniform spatial distribution and the database objects are clustered based on their most selective dimensions. As a result, the approximated verification rate remains under 14% in all data spaces, and the speed-up factors compared to the sequential scan exceed 18 for data spaces with 16, 28, 36 and 40 dimensions. Note that the query selectivity is the

same as in the previous experiment. This test is relevant for the efficiency of our clustering strategy based on access probability.

6.3 Evaluation using real data

The real data used in our experiments has been obtained from a collection of 32-dimensional feature vectors extracted from an image database containing around 60,000 images[3]. The feature vectors have been normalized to the hypercube $[0,1]^{32}$ and a principal component analysis transformation was used against the feature vectors to represent them in a hyper-space where the 32 dimensions were sorted by their importance. To obtain a collection of 10-dimensional points, we selected the first ten dimensions, and generated 32 million points, keeping the initial spatial distribution of the feature vectors. The set of points was then partitioned into 4^{10} (1,048,576) pages. The partitioning was done by recursively dividing the set of points in four sub-partitions, successively using the ten considered dimensions. As a result, each page contained around 32 points, corresponding to 4 kbytes of row data. Each page is decribed by a 10-dimensional extended object representing the bounding box enclosing the points from inside the page. The number of extended objects describing the pages is 4^{10}. According to the spatial distribution of the feature vectors (for the first ten dimensions), the extended objects had different extensions, varying between 0.0001% and 67% of the domain size. We indexed all these extended objects in main memory using our RVA-clustering approach. When an extended object matches the query the corresponding page is accessed from disk. The average extended object selection time was measured based on 10,000 intersection queries addressed to the indexing structure. We conducted several experiments, varying the sizes of the intervals defined by the query objects, in order to control the query selectivity. The experimental results are reported in Table 1. The first column gives the average size of the intervals defined by the query objects. The second column shows the average number of extended objects selected by the query and corresponding to the pages of points that need to be examined. The third column gives the average selection time spent to determine the set of qualifying pages. The forth column indicates the speed-up factor of our approach compared to the sequential scan of the set of extended objects.

As it can be noticed, our indexing technique behaves efficiently, taking advantage of the distribution of the real data. Two aspects are relevant: First, our technique limits the number of pages that need to be read from the external support. This is very important considering that the total number of pages is of 1,048,576. Second, the verification in memory of the extended objects is much faster than the sequential scan (between 37 and 78 times faster) because our technique groups the extended objects based on their most selective dimensions and avoids many useless spatial tests.

The results confirmed our expectations of good behavior against non-uniform real data, also validating the possible application of our technique as clustering support for large collection of points.

[3]The set of feature vectors was provided by Imedia Research Project, INRIA-Rocquencourt, http://www-rocq.inria.fr/imedia/ .

Avg. Query Interval Size	Number of Sel. Pages	Selection Time (ms)	Speed-Up Factor
0.174	17.93	4.74	78.34
0.196	27.32	5.27	73.38
0.221	41.98	7.44	50.47
0.248	144.32	7.84	49.95
0.279	651.80	11.86	34.20
0.315	1243.27	11.41	36.41

Table 1: Experimental Results for Real Data

7 Conclusions

In this paper we presented the RVA-clustering a new approach for answering spatial queries over large collections of extended objects. We defined an approximation model for extended objects and used it to accelerate the verification of the spatial queries and to organize the database in clusters of object approximations. Our clustering strategy, based on access probabilities, behaves efficiently against skewed data and/or skewed queries. Our method copes with large target databases and many dimensions and supports incomplete, heterogeneous, and large extended objects. As future work, we plan to develop an analytical model in order to dynamically establish the clustering parameters and better support the cluster split decision. As clustering support for large collections of hyperspace points, we want to develop methods to efficiently answer more complex similarity queries like ϵ-distance and nearest neighbor queries.

8 Acknowledgments

We wish to specially thank Françoise Fabret[4] who carefully reviewed this report. We also wish to thank Marin Ferecatu from Imedia Research Project[5] who provided us the real data set used in our experiments.

References

[1] N. Beckmann, H.-P. Kriegel, and B. Seeger R. Schneider. The R*-Tree: An Efficient and Robust Access Method for Points and Rectangles. In *Proc. ACM SIGMOD Int. Conf. on Management of Data, Atlantic City, NJ*, 1990.

[4]Member of Caravel Project, INRIA-Rocquencourt, http://www-caravel.inria.fr
[5]INRIA-Rocquencourt, http://www-rocq.inria.fr/imedia/

[2] S. Berchtold, C. Böhm, H. V. Jagadish, H.-P. Kriegel, and J. Sander. Independent Quantization: An Index Compression Technique for High-Dimensional Data Spaces. In *Proc. 16th Int. Conf. on Data Engineering (ICDE), San Diego, CA*, 2000.

[3] S. Berchtold, C. Böhm, and H.-P. Kriegel. The Pyramid-Technique: Towards Breaking the Curse of Dimensionality. In *Proc. Int. Conf. on Management of Data, ACM SIGMOD, Seattle, Washington*, 1998.

[4] S. Berchtold, D. Keim, and H.-P. Kriegel. The x-tree: An index structure for high-dimensional data. *22nd Conf. on Very Large Databases, Bombay, India*, pages 28–39, 1996.

[5] C. Böhm, S. Berchtold, and D. A. Keim. Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases. *ACM Computing Surveys*, 33(3):322–373, 2001.

[6] C. Y. Chan and Y. E. Ioannidis. An Efficient Bitmap Encoding Scheme for Selection Queries. In *SIGMOD 1999, Proc. ACM SIGMOD Int. Conf.on Management of Data, Philadelphia, Pennsylvania, USA*, 1999.

[7] F. Fabret, H.A. Jacobsen, F. Llirbat, J. Pereira, K.A. Ross, and D. Shasha. Filtering Algorithms and Implementation for Very Fast Publish/Subscribe Systems. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, 1997.

[8] C. Faloutsos and P. Bhagwat. Declustering Using Fractals. *PDIS Journal of Parallel and Distributed Information Systems*, pages 18–25, 1993.

[9] V. Gaede and O. Günther. Multidimensional Access Methods. *ACM Computing Surveys*, 30(2):170–231, 1998.

[10] K., J. Goldstein, R. Ramakrishnan, and U. Shaft. When is "nearest neighbor" meaningful? *Lecture Notes in Computer Science*, 1540:217–235, 1999.

[11] N. Katayama and S. Satoh. The SR-Tree: An index Structure for High-Dimensional Nearest Neighbor Queries. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, 1997.

[12] Y. Sakurai, M. Yoshikawa, S. Uemura, and H. Kojima. The A-tree: An Index Structure for High-Dimensional Spaces Using Relative Approximation. In *Proceedings of the 26th VLDB Conference, Cairo, Egypt*, 2000.

[13] R. Weber and S. Blott. An approximation based data structure for similarity search. In *Technical Report 24, ESPRIT Project HERMES (no. 9141), October*, 1997.

[14] R. Weber, H.-J. Schek, and S. Blott. A Quantitative Analysis and Performance Study for Similarity-Search Methods in High-Dimensional Spaces. In *Proceedings of the 24th VLDB Conference, New York, USA*, 1998.

Contents

Unité de recherche INRIA Rocquencourt
Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)

Unité de recherche INRIA Futurs : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38330 Montbonnot-St-Martin (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399